UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/802,586	03/17/2004	Joel David Munter	MP1502	5149

64768 7590 12/13/2007
MARSHALL, GERSTEIN & BORUN, LLP (MARVELL)
233 SOUTH WACKER DRIVE
6300 SEARS TOWER
CHICAGO, IL 60606-6357

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
12/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/802,586	MUNTER ET AL.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

[illegible]

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 September 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated September 4, 2007, responding to the Office action mailed June 4, 2007 provided in the rejection of claims 1-30, wherein claims 1, 3-8, 11-13, 15-19, 22-24, and 26-28 are amended.

Claims 1-30 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Chauvel et al. (Pat. No. US 7,146,613 B2)*, art made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 102(b)

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(b) that form the basis for the rejections under this section made in this office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 11, 13-14, 18-21, and 26-30 are rejected under 35 U.S.C. 102(b) as being anticipated by Chauvel et al. (Pub. No. US 2004/0010785 A1) (hereinafter 'Chauvel-1')

3. **As to claim 11** (Currently Amended) Chauvel-1 discloses in an electronic device, a method of operation, comprising:

- receiving a plurality of non-native instructions (e.g., [0008], the JVM executes the byte-codes just as a processor executes machine code; however, the byte-codes do not directly control the underlying hardware; instead, they are interpreted by the JVM);
- executing the non-native instructions for an initial number of times using an interpreter (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance

characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool'); and

- compiling the non-native instructions into object code after executing the received non-native instructions for said initial number of times using the interpreter ([0063] – the profiling techniques described herein could be adapted to take into account an interpreter-base execution and a JIT one; Just In Time compilers may improve performance due to their sophisticated optimizations created dynamically).

4. **As to claim 13** (Currently Amended) (incorporating the rejection in claim 11), Chauvel-1 discloses the method wherein the method further comprises monitoring said compiling for a compilation requirement employed in determining the initial number of times the received non-native instructions are to be executed using the interpreter before compiling (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance

characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool'); and

- updating a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

5. **As to claim 14** (Original) (incorporating the rejection in claim 11), Chauvel-1 discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for execution requirements of the native instructions; and
- updating execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

6. **As to claim 18** (Currently Amended) , Chauvel-1 discloses an article of manufacture comprising:

- a computer readable medium (e.g., Fig. 1b – element of 16 – HW Platform; [0025], Lines 1-2, 5-11 – a JVM profile is generated for a specific hardware platform); and
- a plurality of instructions designed to implement a runtime manager equipped to receive a plurality of non-native instructions, execute the non-native instructions for an initial number of times using an interpreter (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool'), and invoke a compiler to compile the non-native instructions into object code after executing the received non-native instructions for said initial number of times using the interpreter (e.g., [0063] – the profiling techniques described herein could be adapted to take into account an interpreter-base execution and a JIT

one; Just In Time compilers may improve performance due to their sophisticated optimizations created dynamically).

7. **As to claim 19** (Currently Amended) (incorporating the rejection in claim 18), Chauvel-1 discloses the article wherein the runtime manager is further equipped to determine the initial number of times the received non-native instructions are to be executed before compiling the received non-native instructions (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]).

8. **As to claim 20** (Original) (incorporating the rejection in claim 18), Chauvel-1 discloses the article wherein the runtime manager is further equipped to

- monitor said compiling for a compilation requirement employed in determining the initial number of times received non-native instructions are to be executed before compiling (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]); and

- update a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

9. **As to claim 21** (Original) (incorporating the rejection in claim 18), Chauvel-1 discloses the article wherein the generated object code comprises a plurality of native instructions, and the runtime manager is further equipped to

- monitor execution of the generated object code for execution requirements of the native instructions; and
- update execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

10. **As to claim 26** (Currently Amended), Chauvel-1 discloses a system, comprising:

- a communication interface to receive a plurality of non-native instructions (e.g., [0010], Lines 1-3 – to optimize an application, estimations of execution time or energy consumption are often needed; this is particularly true in the case of mobile devices…; [0061], Lines 1-7 – can be downloaded through a network);
- a storage medium coupled to the communication interface, and having stored therein a plurality of instructions designed to implement a runtime manager equipped to execute the received non-native instructions for an initial number of times using an interpreter (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool'), and invoke a compiler to compile the non-native instructions into object code after executing the received non-

native instructions for said initial number of times using the interpreter (e.g., [0063] – the profiling techniques described herein could be adapted to take into account an interpreter-base execution and a JIT one; Just In Time compilers may improve performance due to their sophisticated optimizations created dynamically); and

- a processor coupled to the storage medium to execute the instructions implementing the runtime manager (e.g., [0008]).

11. **As to claim 27** (Currently Amended) (incorporating the rejection in claim 26), Chauvel-1 discloses the system wherein the runtime manager is further equipped to determine the initial number of times the received non-native instructions are to be executed using the interpreter before invoking the compiler to compile the received non-native instructions (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each

operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool').

12. **As to claim 28** (Currently Amended) (incorporating the rejection in claim 26), Chauvel-1 discloses the system wherein the runtime manager is further equipped to

- monitor said compiling for a compilation requirement employed in determining the initial number of times received non-native instructions are to be executed using the interpreter before compiling (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool'); and
- update a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Fig. 2 – a state diagram describing the

generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

13. **As to claim 29** (Original) (incorporating the rejection in claim 26), Chauvel-1 discloses the system wherein the generated object code comprises a plurality of native instructions, and the runtime manager is further equipped to

- monitor execution of the generated object code for execution requirements of the native instructions (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]); and

- update execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

14. **As to claim 30** (Original) (incorporating the rejection in claim 26), Chauvel-1 discloses the system wherein the communication interface is a wireless communication interface (e.g., [0010], Lines 1-3 – to optimize an

application, estimations of execution time or energy consumption are often needed; this is particularly true in the case of mobile devices...; [0061], Lines 1-7 – can be downloaded through a network).

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. Claims 1-10, 12, 15-17, and 22-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chauvel-1 in view of Chauvel et al. (Pat. No. US 7,146,613 B2) (hereinafter 'Chauvel-2')

16. **As to claim 1** (Currently Amended), Chauvel-1 discloses a method comprising:

- receiving a plurality of non-native instructions (e.g., [0008] - the JVM executes the byte-codes just as a processor executes machine code; however, the byte-codes do not directly control the underlying hardware; instead, they are interpreted by the JVM) in a selected one of a source form and an intermediate form (e.g., [0008] – JAVA™ programs [source form] are compiled into "byte-codes" [an intermediate form]).

Chauvel-1 disclose the profiling techniques described herein could be adapted to take into account an interpreter-based execution and a JIT one, but does not explicitly disclose compiling the plurality of non-native instructions to generate object code for the non-native instructions, wherein compiling the plurality of non-native instructions includes replacing an object code segment from the generated object code with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an amount of energy required to execute the generated object code in a target execution environment.

However, in an analogous art of *Java® DSP Acceleration by Byte-Code Optimization*, Chauvel-2 discloses compiling the plurality of non-native instructions to generate object code for the non-native instructions (e.g., Col. 4, Lines 10 -15 – the hardware interpreter is essentially an on-the-fly interpretation engine that generates native code from byte-codes (non-native instructions); for embedded system in particular, techniques for improving the performance of a software based JVM are needed), wherein compiling the plurality of non-native instructions includes replacing an object code segment from the generated object code with an alternative object code segment if the alternative object code segment (e.g., Fig. 4,–element 420 'Execute on Acceleration Circuitry'; Col. 4, Lines 34-41 - If the certain type of iterative sequence is present, the iterative sequence is replaced with a proprietary code sequence…, and the proprietary code sequences are executed directly by acceleration circuitry; Fig. 1 – illustrating a process flow for implementing an application using a JAVA® Virtual

Machine; Col. 5, Lines 58-62 – The JVM also modifies certain sequences of the byte-code by replacing the selected sequence with a proprietary construct that is executed by acceleration circuitry connected to the processor in order to accelerate execution of application program; Fig. 2; Col. 5, Lines 64-66 – Fig. 2 is a representation of JAVA® byte-code with a proprietary code sequence; Col. 6, Lines 13-18 – Thus, modified byte-code sequence 210 contains one less instruction since two byte-code instructions have been replaced by one proprietary instruction. Furthermore, the proprietary DSP-fmac instruction will be executed on a specialized MAC unit in a faster manner than if the JVM interpreted each byte-code that was replaced) improves at least a selected one of a power level required and an amount of energy required to execute the generated object code in a target execution environment (e.g., Col. 12, Lines 1-11 – many other types of communications systems and computer systems may also benefit from the present invention, particularly those relying of battery power).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chauvel-2 into the Chauvel-1's system to further provide compiling the plurality of non-native instructions to generate object code for the non-native instructions, wherein compiling the plurality of non-native instructions includes replacing an object code segment from the generated object code with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an amount of energy required to execute the generated object code in a target execution environment in Chauvel-1's system.

The motivation is that it would further enhance the Chauvel-1's system by taking, advancing and/or incorporating Chauvel-2's system which offers significant advantages that many other types of communications systems and computer systems may also benefit from the present invention, particularly those relying on batter power as once suggested by Chauvel-2 (e.g., Col. 12, Lines 1-11).

17. **As to claim 2** (Original) (incorporating the rejection in claim 1), Chauvel-1 discloses the method wherein said receiving comprises receiving the non-native instructions in a byte code form (e.g., [0024], Lines 3-7 – the application profile is a byte-code based profile which indicates how many times each operation (such as a byte-code) is used during execution of the application, or in specified parts of the application).

18. **As to claim 3** (Currently Amended) (incorporating the rejection in claim 1), Chauvel-1 discloses compiling comprises analyzing the object code segment for execution power level requirement, and determining whether an alternative object code segment with lower execution power level requirement is available (e.g., Abstract – creating application profiles that indicate the number of execution of each operation in the application and virtual machine profiles which indicate the time/energy consumed by each operation on a particular hardware

platform; an application profile in conjunction with the virtual machine profile can be used to generate time and/or energy estimates for the application; [0062] – to adapt the JVM profile in order to obtain energy consumption estimations of a JAVA™ application; instead of (or in addition to) profiling the execution time on the target, the estimation of energy consumption could be performed using the same principle as for execution; the energy performance (and execution time performance) could be used by the target device for scheduling applications; [0066], Lines 32-35 – acquiring a virtual machine profile that relates power consumption to individual operations; [0010] – to optimize an application, estimations of execution time or energy consumption are often needed; this is particular true in the case of mobile devices, such as smart phones, personal digital assistants, and the like, which have limit energy and processing resources).

19. **As to claim 4** (Currently Amended) (incorporating the rejection in claim 1), Chauvel-1 discloses compiling comprises analyzing the object code segment for execution energy consumption, and determining whether an alternative object code segment with lower execution energy consumption is available (e.g., Abstract – creating application profiles that indicate the number of execution of each operation in the application and virtual machine profiles which indicate the time/energy consumed by each operation on a particular hardware platform; an application profile in conjunction with the virtual machine profile can be used to generate time and/or energy estimates for the application; [0062] – to adapt the

JVM profile in order to obtain energy consumption estimations of a JAVA™ application; instead of (or in addition to) profiling the execution time on the target, the estimation of energy consumption could be performed using the same principle as for execution; the energy performance (and execution time performance) could be used by the target device for scheduling applications; [0066], Lines 32-35 – acquiring a virtual machine profile that relates power consumption to individual operations; [0010] – to optimize an application, estimations of execution time or energy consumption are often needed; this is particular true in the case of mobile devices, such as smart phones, personal digital assistants, and the like, which have limit energy and processing resources).

20. **As to claim 5** (Currently Amended) (incorporating the rejection in claim 1), Chauvel-1 discloses the method wherein

- the method further comprises executing the non-native instructions for an initial number of times using an interpreter (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool'); and

- performing said compiling only after executing the non-native instructions for said initial number of times (e.g., [0063] – the profiling techniques described herein could be adapted to take into account an interpreter-base execution and a JIT one; Just In Time compilers may improve performance due to their sophisticated optimizations created dynamically).

21. **As to claim 6** (Currently Amended) (incorporating the rejection in claim 5), Chauvel-1 discloses the method wherein the method further comprises determining the initial number of times the received non-native instructions are to be executed using the interpreter before performing compiling the received non-native instructions (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]; [0036] – to calculate the application profile, the application is executed with a JVM, which

is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool').

22. **As to claim 7** (Currently Amended) (incorporating the rejection in claim 6), Chauvel-1 discloses the method wherein the method further comprises

- monitoring said compiling for power level required to perform compilation;
- updating a current understanding of power level required for compilation and determining the initial number of times received non-native instructions are to be executed using the interpreter (e.g., [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool') before compiling the received non-native instructions (e.g., [0012] – performance for a specified portion of an application, where the specified portion can include all or part of the application, the executes on a target device via a virtual machine interface is estimated by acquiring an application profile that specifies a number of executions for a plurality of operations used in the specified portion of the application, acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile; [0013]), if said monitoring observes a power level required for compilation to be different from the current understanding (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the

execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

23. **As to claim 8** (Currently Amended) (incorporating the rejection in claim 6), Chauvel-1 discloses the method wherein the method further comprises

- monitoring said compiling for amount of energy required to perform an average compilation;
- updating a current understanding of amount of energy required for an average compilation; and determining initial number of times received non-native instructions are to be executed using the interpreter (e.g., [0036] – to calculate the application profile, the application is executed with a JVM, which is instrumented to count each operation and calculate the profile. The JVM which is used to generate the profile is referred to as the 'profiling tool') before compiling the received non-native instructions, if said monitoring observes an

amount of energy required for compilation to be different from the current understanding (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

24. **As to claim 9** (Original) (incorporating the rejection in claim 1), Chauvel-1 discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for power level required to execute the native instructions; and
- updating power level requirements of selected ones of the native instructions if said monitoring observes power level requirements for the selected ones of the native instructions to be different from current understandings of the

power level requirements of the selected ones of the native instructions (e.g., Fig. 2 – a state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

25. **As to claim 10** (Original) (incorporating the rejection in claim 1), Chauvel-1 discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for amount of energy required to execute the native instructions; and
- updating energy requirements of selected ones of the native instructions if said monitoring observes energy requirements for the selected ones of the native instructions to be different from current understandings of the energy requirements of the selected ones of the native instructions (e.g., Fig. 2 – a

state diagram describing the generation of an application profile; [0029] – the execution of each operation in the application increments a counter associated with the particular operation.; each executed operation is counted until an off command is received; a save results command causes the values of the counters to be stored in file; Fig. 3a – a state diagram describing the generation timing information for a virtual machine profile; Fig. 3b – a state diagram describing the generation energy information for a virtual machine profile; [0045] – the energy consumption data could be based on resources used by the operation and the time of execution; maximum and minimum execution times associated with the operation are maintained as well; upon receiving the save results command, the average energy consumption is calculated for each operation and is stored in the JVM profile).

26. **As to claim 12** (Currently Amended) (incorporating the rejection in claim 11), please refer to claim **6** as set forth accordingly.

27. **As to claim 15** (Currently Amended) , Chauvel-1 discloses an article of manufacture comprising:

- a computer readable medium (e.g., Fig. 1b – element of 16 – HW Platform; [0025], Lines 1-2, 5-11 – a JVM profile is generated for a specific hardware platform).

Chauvel-1 disclose the profiling techniques described herein could be adapted to take into account an interpreter-based execution and a JIT one, but does not explicitly disclose a plurality of instructions designed to implement a compiler to compile non-native instructions to generate object code for the non-native instructions, and replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code.

However, in an analogous art of *Java® DSP Acceleration by Byte-Code Optimization*, Chauvel-2 discloses a plurality of instructions designed to implement a compiler to compile non-native instructions to generate object code for the non-native instructions (e.g., Col. 4, Lines 10 -15 – the hardware interpreter is essentially an on-the-fly interpretation engine that generates native code from byte-codes (non-native instructions); for embedded system in particular, techniques for improving the performance of a software based JVM are needed), and replace an object code segment with an alternative object code segment if the alternative object code segment (e.g., Fig. 4,–element 420 'Execute on Acceleration Circuitry'; Col. 4, Lines 34-41 - If the certain type of iterative sequence is present, the iterative sequence is replaced with a proprietary code sequence…, and the proprietary code sequences are executed directly by acceleration circuitry; Fig. 1 – illustrating a process flow for implementing an application using a JAVA® Virtual Machine; Col. 5, Lines 58-62 – The JVM also modifies certain sequences of the byte-code by replacing the selected sequence with a proprietary construct that is executed by acceleration circuitry connected to the processor in order to accelerate execution of application program; Fig. 2; Col. 5, Lines 64-66 – Fig. 2 is a representation of JAVA® byte-code with a proprietary code sequence; Col. 6, Lines 13-18 – Thus, modified byte-code sequence 210 contains one less instruction since two byte-code instructions have been replaced by one proprietary instruction. Furthermore, the proprietary DSP-fmac instruction will be executed on a specialized MAC unit in a faster manner than if the JVM interpreted each byte-code that was replaced) improves at least a selected one of a power level required and an energy required to execute the generated object code (e.g., Col. 12, Lines 1-11 – many other types of communications systems and computer systems may also benefit from the present invention, particularly those relying of battery power).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chauvel-2 into the Chauvel-1's system to further provide a plurality of instructions designed to implement a compiler to compile non-native instructions to generate object code for the non-native instructions, and replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code in Chauvel-1's system.

The motivation is that it would further enhance the Chauvel-1's system by taking, advancing and/or incorporating Chauvel-2's system which offers significant advantages that many other types of communications systems and computer systems may also benefit from the present invention, particularly those relying on batter power as once suggested by Chauvel-2 (e.g., Col. 12, Lines 1-11).

28. **As to claim 16** (Currently Amended) (incorporating the rejection in claim 15), please refer to claim **3** as set forth accordingly

29. **As to claim 17** (Currently Amended) (incorporating the rejection in claim 15), please refer to claim **4** as set forth accordingly.

30. **As to claim 22** (Currently Amended) , Chauvel-1 discloses a system, comprising:

- a storage medium having stored therein a plurality of instructions implementing a compiler to compile non-native instructions to generate object code for the non-native instructions, and replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code; and
- a processor coupled to the storage medium to execute the instructions implementing the compiler (e.g., Abstract – creating application profiles that indicate the number of execution of each operation in the application and virtual machine profiles which indicate the time/energy consumed by each operation on a particular hardware platform; an application profile in conjunction with the virtual machine profile can be used to generate time

and/or energy estimates for the application; [0062] – to adapt the JVM profile in order to obtain energy consumption estimations of a JAVA™ application; instead of (or in addition to) profiling the execution time on the target, the estimation of energy consumption could be performed using the same principle as for execution; the energy performance (and execution time performance) could be used by the target device for scheduling applications; [0066], Lines 32-35 – acquiring a virtual machine profile that relates power consumption to individual operations; [0010] – to optimize an application, estimations of execution time or energy consumption are often needed; this is particular true in the case of mobile devices, such as smart phones, personal digital assistants, and the like, which have limit energy and processing resources).

Chauvel-1 disclose the profiling techniques described herein could be adapted to take into account an interpreter-based execution and a JIT one, but does not explicitly disclose a storage medium having stored therein a plurality of instructions implementing a compiler to compile non-native instructions to generate object code for the non-native instructions, and replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code.

However, in an analogous art of *Java® DSP Acceleration by Byte-Code Optimization*, Chauvel-2 discloses a storage medium having stored therein a plurality of instructions implementing a compiler to compile non-native instructions to generate object code for the non-native instructions (e.g., Col. 4, Lines 10 -15 – the hardware interpreter is essentially an on-the-fly interpretation engine that generates native code from byte-codes (non-native instructions); for embedded system in particular, techniques for improving the performance of a software based JVM are needed), and replace an object code segment with an alternative object code segment if the alternative object code segment (e.g., Fig. 4,–element 420 'Execute on Acceleration Circuitry'; Col. 4, Lines 34-41 - If the certain type of iterative sequence is present, the iterative sequence is replaced with a proprietary code sequence…, and the proprietary code sequences are executed directly by acceleration circuitry; Fig. 1 – illustrating a process flow for implementing an application using a JAVA® Virtual Machine; Col. 5, Lines 58-62 – The JVM also modifies certain sequences of the byte-code by replacing the selected sequence with a proprietary construct that is executed by acceleration circuitry connected to the processor in order to accelerate execution of application program; Fig. 2; Col. 5, Lines 64-66 – Fig. 2 is a representation of JAVA® byte-code with a proprietary code sequence; Col. 6, Lines 13-18 – Thus, modified byte-code sequence 210 contains one less instruction since two byte-code instructions have been replaced by one proprietary instruction. Furthermore, the proprietary DSP-fmac instruction will be executed on a specialized MAC unit in a faster manner than if the JVM interpreted each byte-code that was replaced) improves at least a selected one of a power level required and an energy required to execute the generated object code (e.g., Col. 12, Lines 1-11 – many other types of communications systems and computer systems may also benefit from the present invention, particularly those relying of battery power).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chauvel-2 into the Chauvel-1's system to further provide a storage medium having stored therein a plurality of instructions implementing a compiler to compile non-native instructions to generate object code for the non-native instructions, and replace an object code segment with an alternative object code segment if the alternative object code segment improves at least a selected one of a power level required and an energy required to execute the generated object code in Chauvel-1's system.

The motivation is that it would further enhance the Chauvel-1's system by taking, advancing and/or incorporating Chauvel-2's system which offers significant advantages that many other types of communications systems and computer systems may also benefit from the present invention, particularly those relying on batter power as once suggested by Chauvel-2 (e.g., Col. 12, Lines 1-11).

31. **As to claim 23** (Currently Amended) (incorporating the rejection in claim 22), Chauvel-1 discloses the system wherein said compiler analyzes the object code segment for execution power level requirement, and determining whether an alternative object code segment with lower execution power level requirement is available (e.g., Abstract – creating application profiles that indicate the number

of execution of each operation in the application and virtual machine profiles which indicate the time/energy consumed by each operation on a particular hardware platform; an application profile in conjunction with the virtual machine profile can be used to generate time and/or energy estimates for the application; [0062] – to adapt the JVM profile in order to obtain energy consumption estimations of a JAVA™ application; instead of (or in addition to) profiling the execution time on the target, the estimation of energy consumption could be performed using the same principle as for execution; the energy performance (and execution time performance) could be used by the target device for scheduling applications; [0066], Lines 32-35 – acquiring a virtual machine profile that relates power consumption to individual operations; [0010] – to optimize an application, estimations of execution time or energy consumption are often needed; this is particular true in the case of mobile devices, such as smart phones, personal digital assistants, and the like, which have limit energy and processing resources).

32. **As to claim 24** (Currently Amended) (incorporating the rejection in claim 22), Chauvel-1 discloses the system wherein said compiler analyzes the object code segment for execution energy consumption, and determining whether an alternative object code segment with lower execution energy consumption is available (e.g., Abstract – creating application profiles that indicate the number of execution of each operation in the application and virtual machine profiles which indicate the time/energy consumed by each operation on a particular hardware

platform; an application profile in conjunction with the virtual machine profile can be used to generate time and/or energy estimates for the application; [0062] – to adapt the JVM profile in order to obtain energy consumption estimations of a JAVA™ application; instead of (or in addition to) profiling the execution time on the target, the estimation of energy consumption could be performed using the same principle as for execution; the energy performance (and execution time performance) could be used by the target device for scheduling applications; [0066], Lines 32-35 – acquiring a virtual machine profile that relates power consumption to individual operations; [0010] – to optimize an application, estimations of execution time or energy consumption are often needed; this is particular true in the case of mobile devices, such as smart phones, personal digital assistants, and the like, which have limit energy and processing resources).

33. **As to claim 25** (Original) (incorporating the rejection in claim 22), Chauvel-1 discloses the system wherein the apparatus further comprises a wireless communication interface to receive the non-native instructions (e.g., [0010], Lines 1-3 – to optimize an application, estimations of execution time or energy consumption are often needed; this is particularly true in the case of mobile devices...; [0061], Lines 1-7 – can be downloaded through a network).

Conclusion

34. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is

571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

BCW BW

December 7, 2007

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER